Eduardo Xavier da Silva

Orchidea:
Uma orquestra de dispositivos Android

São João Del-Rei

2015

Eduardo Xavier da Silva

Orchidea:
Uma orquestra de dispositivos Android

Monografia apresentada como requisito da disciplina de Projeto Orientado em Computação II do Curso de Bacharelado em Ciência da Computação da UFSJ.

Universidade Federal de São João Del-Rei – UFSJ

Bacharelado em Ciência da Computação

Orientador: Flávio Luiz Schiavoni

São João Del-Rei

2015

Eduardo Xavier da Silva
Orchidea:
Uma orquestra de dispositivos Android/ Eduardo Xavier da Silva. – São João Del-Rei, 2015-
38 p. : il. (algumas color.) ; 30 cm.

Orientador: Flávio Luiz Schiavoni

Monografia (Graduação) – Universidade Federal de São João Del-Rei – UFSJ
Bacharelado em Ciência da Computação, 2015.

1. Android. 2. Aplicativo. 3. Música. I. Flávio Luiz Schiavoni. II. Universidade Federal de São João Del Rei. III. Orchidea Uma orquestra de dispositivos Android.
CDU 02:141:005.7

Eduardo Xavier da Silva

Orchidea:
Uma orquestra de dispositivos Android

Monografia apresentada como requisito da disciplina de Projeto Orientado em Computação II do Curso de Bacharelado em Ciência da Computação da UFSJ.

Trabalho aprovado. São João Del-Rei, 11 de dezembro de 2015:

Flávio Luiz Schiavoni
Orientador

Professor
Elder José Reioli Cirilo

Professor
Dárlinton Barbosa Feres Carvalho

São João Del-Rei
2015

Agradecimentos

Um muito obrigado vai para toda a equipe do Android, especialmente aqueles de vocês que já contribuíram para o *Android developer Google Groups*, onde há um grande apoio para a criação de novos aplicativos.

Um muito obrigado também aos desenvolvedores de Pure Data como Miller Puckette, que veio a ser utilizado para fazer o libpd para Android por Peter Brinkmann, onde esta aplicação colaborou com o desenvolvimento da ideia.

Queria deixar um agradecimento especial para o meu orientador Flávio Luiz Schiavoni, que foi não só o idealizador do projeto como também me ajudou muito no desenvolvimento deste no decorrer deste ano e também à um ex colega de classe Matheus Felipe Marquês da Silva que me deu apoio no desenvolvimento da aplicação.

Sumário

Lista de ilustrações

1 Introdução

Com o desenvolvimento da tecnologia e a demanda do mercado, surgiu um novo paradigma da computação. A chamada computação móvel baseia-se em um aparelho móvel que não apenas faz ligações telefônicas como qualquer outro aparelho desenvolvido até o momento, mas possui diversas outras funcionalidades. Os primeiros smarthphones eram um telefone inteligente baseado na combinação entre celulares e agendas eletrônicas. Estes dispositivos possuem uma tecnologia mais avançada e possuem um sistema operacional, no caso o que iremos trabalhar neste documento é o sistema operacional Android que é baseado na arquitetura Linux e foi desenvolvido pela empresa de tecnologia Google.

A plataforma móvel Android tornou-se atualmente a mais comum entra as centenas de milhões de dispositivos moveis em mais de 190 países do mundo. Com esta grande utilização surgiu uma grande diversidade de usuários e, com isto, criou-se a necessidade de aplicativos que atendam a esta nova gama de usuários. Um tipo de aplicativo utilizado por alguns usuários é a aplicação mobile envolvida diretamente com a música.

A partir desta plataforma surgiu a ideia da implementação de um aplicativo móvel que fizesse com que os usuários estivessem fazendo parte de uma orquestra, em que cada usuário tocaria um instrumento desta orquestra, e todos os usuários interagiriam entre si, tocando instrumentos diversos e gerando sons para todos conectados neste aplicativo.

Um outro meio de produzir sons utilizando a tecnologia é conhecido como laptop orquestra, que veio anterior a orquestra mobile, onde a partir dos projetos nesta área e o desenvolvimento da tecnologia surgiu a orquestra mobile, a laptop orquestra consiste em utilizar computadores interligados, geralmente via rede, para produzir sons.

1.1 Motivação

Com a evolução da tecnologia móvel, os aparelhos celulares e "smartphones" se tornaram computadores portáteis presentes na vida cotidiana de muitas pessoas. O objetivo inicial destes dispositivos era a comunicação e hoje os mesmos cumprem este papel ultrapassando a simples comunicação de voz e permitindo a comunicação por outras mídias. A expansão de tais limites comunicacionais foi possível graças à evolução da capacidade dos dispositivos principalmente no que tange à conectividade e à capacidade de processamento aliada à possibilidade de manipulação de materiais multimídia.

A partir desta expansão surgiu a possibilidade de implementação de aplicativos que transformam dispositivos móveis em instrumentos músicais. Apesar de vários aplicativos músicais serem individuais ou focados para um único usuário, é possível integrar vários

destes aplicativos e usuários em uma orquestra de instrumentos digitais.

1.2 Objetivo

O objetivo deste projeto é conceituar algumas ferramentas disponíveis sobre computação músical e aplicá-las na implementação de um aplicativo mobile, que tem como principal objetivo provar que a partir de uma serie de ferramentas disponíveis pode se criar um aplicativo mobile simples e de fácil implementação.

1.3 Trabalhos Relacionados

Embora haja uma série de ferramentas para desenvolvimento de aplicações músicais em computadores desktop, não existem atualmente tantas aplicações disponíveis que oferecem uma estrutura simples para a leitura de sensores disponíveis em dispositivos móveis, voltada a área músical. Esta pesquisa se baseia em trabalhos anteriores que permitem a interatividade complexa através da utilização de vários sensores e a partir destes, gerar diversos sons diferentes em diversos dispositivos Android diferentes.

1.3.1 Dispositivos móveis

Um dos pioneiros na área de desenvolvimento de um instrumento físico utilizando um telefone celular foi Greg Schiemer (SCHIEMER; HAVRYLIV, 2005) em seu Pocket Gamelan, onde ele utilizou o bluetooth como a interação dos dispositivos e como o mecanismo para o envio da música via uma ligação sem fio. Neste trabalho o intuito é utilizar a rede Wi-Fi como o mecanismo para o envio da música, onde iremos enviá-la via Multicast 1.4.5 que será descrito mais adiante no documento.

Figura 1: Mopho

Estudantes da universidade de Stanford (OH et al., 2010) desenvolveram um aplicativo semelhante ao descrito neste trabalho como pode ser visto na Figura 1, para uma orquestra de dispositivos moveis baseado na plataforma do IOS como a sua base de desenvolvimento.

A mesma abordagem descrita neste trabalho pode ser visto no trabalho Shamus (ESSL; ROHS, 2007), que tem uma abordagem baseada em sensores para transformar dispositivos móveis em instrumentos músicais. A ideia principal é utilizar o dispositivo móvel como um instrumento independente de outro dispositivo, fazendo com que cada dispositivo torne se um instrumento da orquestra. Os sensores que o Shamus mais utiliza são o acelerômetros e o magnetômetros para gerar os sons, uma utilização um tanto diferente da qual este trabalho utilizara, onde a ideia deste trabalho consiste em utilizar os sensores touchscreen para a geração de sons, alternando-se em diversos instrumentos a quais serão implementados.

Existem diversas maneiras de produzir sons com dispositivos móveis uma destas pode ser vista no trabalho (**??**), onde o autor comenta uma nova maneira de produzir sons, utilizando ferramentas do celular que geralmente não são utilizadas para tal aplicação, como câmera, sensores de movimento da câmera, sensores de brilho e coisas do tipo, gerando um novo contexto músical, onde os objetos e detalhes do ambiente geram um novo estilo músical. Este novo tipo de contexto músical será pesquisado em um trabalho futuro envolvido com o trabalho descrito neste trabalho, onde os usuários poderão interagir com o ambiente em diversos lugares diferentes, tendo de estar apenas na mesma rede, onde sera feito e empacotamento e o envio de dados. Uma outra maneira de fazer sons via dispositivo mobile pode ser vista,(**??**) em que o usuário podia fazer o uso do acelerômetro, informações de orientação calculada usando uma variedade de sensores microfone, câmera (frente e verso), do magnetômetro que integra a extração de características de sinais de áudio e vídeo para simplificar a sua utilização em diversos contextos.

Um trabalho que seguiu uma ideia semelhante ao desenvolvida neste trabalho é (WANG; ESSL; PENTTINEN, 2008) em que há um repertorio baseado em um conjunto de dispositivos moveis como o instrumento principal da orquestra. Este trabalho cita uma ferramenta que pode ser utilizada por mais de uma duzia de usuários que servem para a composição e performance de um repertorio totalmente dedicado a área músical. Com o uso deste projeto, já foram feitos alguns concertos músicais com até oito músicas, sendo estas músicas, algumas composições e outras algumas improvisações feitas pelo usuário.

Um outro autor que desenvolveu algo semelhante ao apresentado neste trabalho é Atau Tanaka (TANAKA, 2004), que desenvolveu um software baseado em hardware de rede para produzir música em grupo, o sistema que foi desenvolvido explora redes sem fio ad-hoc e a mobilidade de dispositivos moveis para uma comunidade de usuários poderem criar uma única peça de música.

1.3.2 Laptop Orquestra

Entre os trabalhos relacionados é importante citar os grupos músicais que utilizam outro meio de produzir sons utilizando a tecnologia conhecidos como Laptop orquestra. As

Laptop Orchestras vieram anteriormente à orquestra mobile e a partir dos projetos nesta área e o desenvolvimento da tecnologia surgiu a orquestra mobile. A laptop orquestra consiste em utilizar computadores interligados, geralmente via rede, para produzir sons. Um dos pioneiros nessa área Dan Trueman (TRUEMAN, 2007) fez sua primeira orquestra utilizando um conjunto de quinze instrumentos baseados em computadores portáteis em 2005, uma orquestra conhecida como ''Princeton Laptop Orchestra (PLOrk)", apresentada na Figura 2. Dan Trueman compara uma orquestra tradicional com uma orquestra produzida por laptops, onde em suas comparações mesmo a orquestra de laptops sendo um tanto quanto diferente, pode se dizer que o seu objetivo não se difere em nada de uma orquestra tradicional.

Figura 2: PLOrk

Indo além do estudo de Dan Trueman, outras características da tecnologia foram exploradas por este modelo o "Orquestra Stanford Laptop (SLOrk)" (WANG et al., 2009), onde além de explorar os computadores este também explora a utilização de smartphones em suas orquestra. Este projeto aproveita a precisão dos computadores, as possibilidades de produção de infinitos sons e as varias maneiras diferentes que pode-se produzir sons utilizando um computador. Já a "The Carnegie Mellon Laptop Orchestra (CMLO)" (DANNENBERG; SCAVACO; ANG I. AVRAMOVIC, 2007), apresenta uma orquestra que visa explorar o envio das músicas via rede sem fio.

Inspirado pelos trabalhos de (TRUEMAN, 2007) e (WANG et al., 2009) surgiu a ''Orquestra de Laptop Linux" conhecida como L^2Ork (BUKVIC et al., 2010), a primeira orquestra baseada no sistema operacional Linux. Este projeto manteve a compatibilidade com os seus precursores Princeton PLOrk e SLOrk de Stanford, mas como um foco maior na acessibilidade, flexibilidade e corte de custos, mas sem alterar a qualidade do som.

Além destes existe o ''Collaborative Improvisation, and Laptop Ensembles" (LEE et al., 2011) conhecido como LOLC, que baseia-se em textos improvisados e colaborativos, e cuja aplicação visa ser utilizada não apenas por programadores, mas também por não-programadores e músicos. A LOLC não utiliza linguagem de programação para a geração de sons sendo a mesma feita a partir de uma expressão matemática de uma única linha para a geração do som.

1.4 Referencial Teórico

Nesta seção do documento foi conceituado as principais ferramentas e teorias que serão utilizados no trabalho descrito neste documento, baseando se em outros trabalhos já desenvolvidos utilizando tais ferramentas.

1.4.1 Android

O Android é uma plataforma para tecnologia móvel, envolvendo um pacote com programas para celulares, já com um sistema operacional, middleware, aplicativos e interface do usuário (PEREIRA; SILVA,). A plataforma Android foi desenvolvida com base no sistema operacional Linux e é composta por um conjunto de ferramentas que atua em todas as fases do desenvolvimento do projeto, desde a execução até a criação de softwares específicos.

A plataforma Android foi construído com a intenção de permitir aos desenvolvedores criar aplicações móveis que possam tirar toral proveito do que um aparelho portátil possa oferecer, no caso deste trabalho, tirar proveito das diversas maneiras possíveis de gerar sons, baseando se não apenas nos sensores touchscreen, mas visando outras maneiras, como por exemplo a utilização da câmera.

Os aplicativos Android são desenvolvidos utilizando um ou mais componentes básicos, activities, services, content providers e messages. Uma das principais caracteristicas do desenvolvimento de aplicações android são os recursos que possibilitam a separação da interface com a suas funcionalidades (BURNETTE, 2010).

Activities representam a tela associada a um aplicativo. Uma aplicação pode ter uma ou mais activities.

Os services são rotinas que são executados em paralelo utilizando uma thread principal. Este permite o desenvolvimento de ações em segundo plano, sem bloquear a execução da thread principal e interação com essa aplicação.

Os content providers são usados para compartilhar dados entre as aplicações. A partilha de dados é feita através de arquivos, bancos de dados ou outros meios. Uma alternativa para os fornecedores de conteúdos é o uso de comunicação entre os processos, no caso deste aplicativo, utilizando multicast 1.4.5.

Os aplicativos do Android são desenvolvidos principalmente usando o Eclipse IDE com Android Development Tools (ADT) plug-in. SDK e emuladores Android, são as ferramentas necessárias para o desenvolvimento de aplicativos.

1.4.2 GUI - Android

O sistema operacional Android utilizado em dispositivos smartphones tem um hardware limitado e uma tela de pequeno porte, mas geralmente são equipados com um grande número de sensores e dispositivos de comunicação, como um microfone, wi-fi e chips Bluetooth, receptor GPS, única ou múltipla tela touchscreen, sensores de inclinação, câmera e assim por diante. A fim de otimizar a gestão de todos esses recursos e lidar com as limitações do hardware, o sistema Android implementa um modelo de processo multithread em que apenas uma única thread pode acessar a interface de usuário, enquanto as outras funcionalidades são executadas em segundo plano (AMALFITANO; FASOLINO; TRAMONTANA, 2011).

Um aplicativo Android é composto de vários tipos de componentes de Java instanciados em tempo de execução (ou seja, Activities, Services, receptores de Multicast e conteúdo Fornecido), onde os componentes são cruciais para desenvolver a interface de usuário de uma aplicação. A activity é o componente, responsável por apresentar uma interface de usuário visual para cada tarefa. As tarefas inclui geralmente uma ou várias classes de activities que estendem a classe base do desenvolvimento. A interface de usuário mostra cada atividade na tela e é construída usando outras classes de estrutura, tais como View, ViewGroup, Widget, Menu, diálogos, etc.

1.4.3 Broadcast

Broadcasting é um protocolo de envio de mensagens, que utiliza um padrão de envio de mensagens diferente do Multicasting 1.4.5, este não verifica quais nós da rede necessita receber a mensagem, este utiliza do envio para todos os nós conectados a esta rede (KAASHOEK et al., 1989).

1.4.4 Unicast

Unicasting é um outro protocolo de envio de mensagens, que envia apenas uma mensagem de cada vez, sendo que esta terá apenas um destinatário por vez (AALTONEN; KARVO; AALTO, 2002), sendo muito pouco eficaz para o projeto descrito neste documento, onde é necessário o envio para diversos usuários simultaneamente.

1.4.5 Multicast

Novas aplicações emergentes como vídeo conferência, computação distribuída, seminários e alguns aplicativos Android que necessitam do envio de algum dado, como o descrito neste documento são alguns exemplos de processos que necessitam de comunicação multi ponto. Existem três métodos fundamentais para transmissão de dados em uma rede: unicast, broadcast e multicast. Tráfegos unicast são trocados normalmente entre

hosts específicos, como um computador pessoal e um servidor Web. Tráfegos broadcast são enviados a todos os usuários de uma rede. Tráfegos multicast são um exemplo de tráfego destinado a apenas uma parte específica de usuários em uma rede (CAMPOS,).

Multicasting é um método de transmissão de um pacote de dados para múltiplos destinos ao mesmo tempo. Durante uma transmissão Multicast, o transmissor envia os pacotes de dados somente uma vez, ficando a cargo dos receptores captarem esta transmissão e reproduzi-la. Esta técnica diminui consideravelmente o tráfego em diversas situações, como por exemplo, quando vários clientes estão a assistir a uma transmissão de um jogo de futebol, propagado por um servidor (COSTA,).

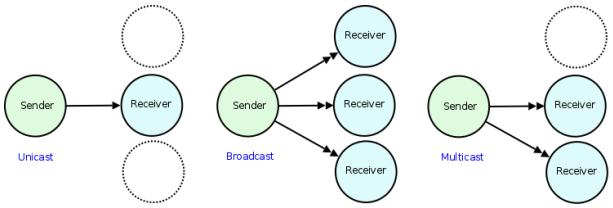

Figura 3: Protocolos de rede

Neste trabalho, ao se analisar a aplicabilidade dos mecanismos de transmissão de dados em uma rede, conforma apresentado na Fig. 3, foi escolhido o multicast como o mecanismo para esta transmissão, pois em relação aos demais, este se encaixa melhor no caso de uso do aplicativo, pois não utiliza a transmissão de dados para apenas um elemento, como no unicast, nem a propagação dos pacotes para todos os elementos da rede, como no broadcast, sendo assim feita a escolha pelo multicast, que envia os pacotes apenas para um determinado grupo da rede, no caso deste trabalho será enviado um pacote de áudio utilizando o protocolo OSC 1.4.9.

1.4.6 libpd

Na etapa de desenvolvimento, necessitou de uma ferramenta de geração sonora e estruturas músicais, para tal tarefa foi escolhida a biblioteca libpd (BRINKMANN PETER KIRN,), onde esta extrai a funcionalidade do Pure Data 1.4.7 e o torná disponível como um callback de processamento de áudio.

O desenvolvimento desta biblioteca libpd para o sistema operacional Android, começou em 2010, onde na época os aplicativos tinham de ser escrito em Java e não haviam a capacidade de a APIs de entrada e saída por meio de código escrito em C. Onde a partir desta limitação surgiu um wrapper Java de Pure Data que porporcionava uma pequena API de processamento de sinal e passagem de mensagens, assim podendo o libpd ser utilizado pelo sistema operacional Android.

1.4.7 Pure Data

Pure data é um ambiente de música em computadores com o funcionamento em tempo real (PUCKETTE,). Pure data é uma linguagem de patching baseada em telas que pode imitar as modalidades de um sintetizador analógico patchable.

1.4.8 Sound Pool e Media Player

Outra ferramenta considerada para a geração de som, foi utilizar classes nativas da linguagem Java, Sound Pool e Media Player são algumas dessas classes, Sound Pool é mais eficaz em relação ao Media Player, na geração de curtos sons e o Media Player e mais eficaz para grandes arquivos sonoros, mas esta aplicação necessita principalmente de sintetização de áudios, onde o libpd acaba sendo mais eficaz.

1.4.9 Protocolo OSC

Open Sound Control (OSC) é um formato de conteúdo de mídia digital para envio de mensagens de conteúdo músical em fluxo de tempo real. Basicamente controla o áudio de qualquer informação baseada no tempo relacionado com um fluxo de áudio que não seja o próprio componente de áudio (SCHMEDER; FREED; WESSEL, 2010). Este será utilizado para enviar os áudios produzidos por cada um dos dispositivos conectados na orquestra.

Mensagens OSC são sequências de quadros definidos com relação a um certo tempo chamado de timetag. Os quadros são chamados de bundles. Onde são um conjunto de um certo número de mensagens, cada destas representam o estado de um sub-fluxo, no timetag. Os sub-fluxos são rotulados com uma seqüência de caracteres legíveis chamados de endereço. Em uma mensagem, o endereço está associado a um vetor de tipos de dados primitivos que incluem codificações de 32 bits binários para inteiros, números reais e texto Figura 4.

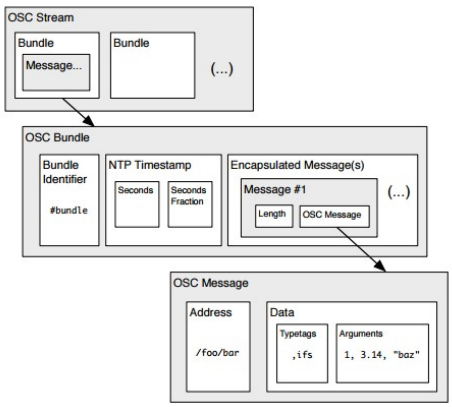

Figura 4: Formato das mensagens OSC

1.4.10 Sensores Touchscreen

Sensores Touchsreen é a forma mais versátil e eficiente para detectar toque humano nos dias atuais (XU; BAI; ZHU, 2012). O touchscreen é a interface de usuário principal dos smartphones. Quando se toca no touchscreen, o display de hardware e firmware irá relatar as coordenadas dos eventos para o sistema operacional do Android e juntamente com a aplicação irá executar alguma ação, dependendo da entrada do usuário.

2 Definição de Arquitetura

Baseado na experiência de grupos que já trabalham o uso de celulares como instrumentos para práticas músicais colaborativas, este trabalho inicia-se propondo uma arquitetura de sistema para desenvolver uma orquestra de dispositivos. O modelo arquitetural escolhido, que pode ser visto na figura 5, utiliza uma arquitetura em camadas para a definição de cada instrumento na orquestra. Usando o modelo pode-se descrever os padrões que são usados nas camadas propostas mantendo um baixo acoplamento no sistema e permitindo a escolha de tecnologias específicas para a implementação de cada funcionalidade. Esse modelo ajuda a entender a aplicação, facilitando as alterações, e a comunicar claramente as intenções do projeto.

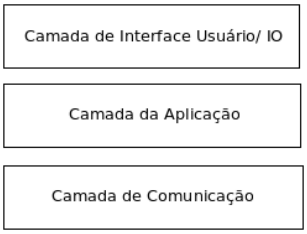

Figura 5: Arquitetura Orchidea

A seguir, apresentaremos cada camada explicitando os componentes de software previsto para sua implementação.

2.1 Camada de Interface Usuário/IO

Na arquitetura proposta, a Camada de Interface Usuário/IO é responsável por garantir a interação do usuário com o sistema. Para isto, ela deve mapear os sensores do smarthphone e responder a entrada do usuário tanto graficamente em sua GUI quanto sonoramente por meio da emissão de sons. Por esta razão, esta camada possui três componentes: Sensores, GUI e Sintetizador, conforme ilustrados na figura 6.

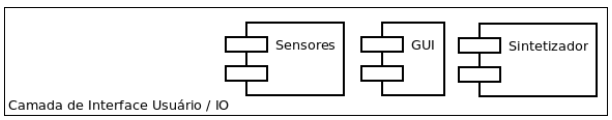

Figura 6: Camada de Interface Usuário / IO

Sensores é o responsável por mapear a interação do usuário e gerar eventos para o Gerenciador de Eventos na Camada de Aplicação. Um instrumento específico irá utilizar um ou mais sensores do dispositivo, o que implica em registrar-se junto a este componente

a escuta destes sensores. É importante notar que nem todo instrumento irá utilizar todos os sensores disponíveis em um aparelho e que a utilização de um instrumento específico implica em um ou mais sensores estarem disponíveis em um determinado aparelho. Também é responsabilidade deste componente definir as restrições (*constraints*) os sensores pois diferentes dispositivos podem possuir diferentes limitações. Assim, mais do que verificar um toque na tela em determinada posição, é necessário que o instrumento faça um mapeamento deste evento em relação ao tamanho da tela de maneira a gerar um evento com significado semântico podendo afirmar, por exemplo, qual tecla do piano foi pressionada por este toque na tela.

A GUI é responsável por refletir ao usuário a utilização do instrumento. É ideal que a mesma receba eventos com significa semântico do Gerenciador de Eventos de forma que a refletir um evento do ambiente. Novamente, tal componente deve ser responsável por conhecer as restrições do dispositivo utilizado para garantir uma precisa representação do evento.

O último componente desta camada é o Sintetizador. Este componente também deverá receber eventos do Gerenciador de Evento mas o mesmo deverá refletir todos os eventos ocorridos no ambiente. Isto inclui tanto eventos locais quanto eventos gerados em outros dispositivos e recebidos pela rede. Por esta razão, o Sintetizador não depende apenas do instrumento escolhido pelo usuário mas da definição de eventos para todos os instrumentos que comporão a orquestra.

2.2 Camada de Aplicação

A Camada de Aplicação é responsável por gerenciar os eventos que ocorrem no aplicativo. Esta possui um componente chamado Gerenciador de Eventos que é responsável local pela troca de mensagens entre todos os demais componentes do sistema, conforme apresentado na Figura 7.

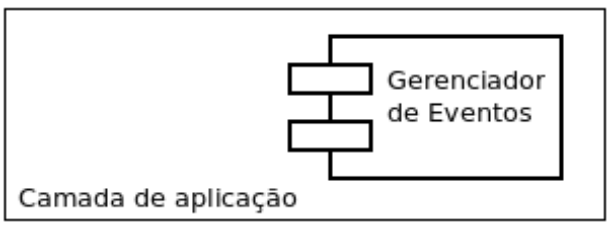

Figura 7: Camada de Aplicação

Gerenciador de Eventos é o centro da aplicação e funciona da mesma maneira para todos os instrumentos desenvolvidos comunicando-se tanto com a Camada de Interface Usuário / IO quanto com a Camada de Comunicação. Ao receber um evento dos Sensores, o Gerenciador de Evento disparará uma mensagem de rede que sinaliza ao ambiente o

evento local e disparará um evento para que a GUI reflita a interação do usuário. Desta maneira, o desenvolvimento de um Instrumento possui um baixo acoplamento entre seus Sensores e sua GUI o que permite um desenvolvimento mais modularizado destes componentes. Ao receber uma mensagem de rede, o Gerenciador de Evento comunica esta mensagem ao Sintetizador que será responsável por refletir sonoramente um evento do ambiente. Este componente deverá ter internamente um buffer circular para a sincronização das mensagens recebidas pela rede.

2.3 Camada de Comunicação

A Camada de Comunicação é responsável pela comunicação em rede. Para realizar esta tarefa foi definido 2 componentes, envio de mensagens e empacotador/desempacotador, ilustrados na figura 8.

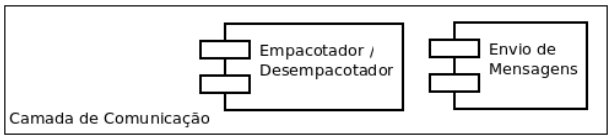

Figura 8: Camada de Comunicação

O Empacotador/Desempacotador é o responsável por definir as mensagens em um formato de rede de maneira a garantir a compatibilidade entre as mensagens do ambiente. Na arquitetura de referência do protocolo TCP/IP, este componente definirá um protocolo de aplicação, especificando formatos de mensagens de maneira que aplicações distintas possam se comunicar por meio de um protocolo comum. Desta maneira, a definição de um instrumento também implica em definir quais mensagens o mesmo terá e como as mesmas serão empacotadas para a distribuição em rede.

O Envio de Mensagens é o responsável por enviar e receber as mensagens para / de os demais dispositivos conectados na rede. Tal componente, na arquitetura de referência do protocolo TCP/IP, possuirá uma implementação de um protocolo de transporte que garanta a comunicação em rede das mensagens já empacotadas.

3 Decisões de Projeto

Este trabalho consiste em fazer uma aplicação músical para dispositivos móveis que rodam sobre a plataforma Android, trabalho este denominado “Orchidea, uma orquestra de dispositivos Android“, este nome veio do intuito do aplicativo ser utilizado para fazer uma orquestra de dispositivos móveis utilizando esta aplicação, cada um destes dispositivos conectados a uma mesma rede e cada um produzindo um som sintetizado de um determinado instrumento.

O projeto pode ser subdivido em três etapas, a primeira etapa consiste na produção e síntese de áudio, a segunda etapa consiste em enviar de comandos para os demais dispositivos conectados a rede para síntese distribuída, a terceira etapa consiste no desenvolvimento da interface de usuário e mapeamento dos sensores touchscreen.

3.1 Primeira Etapa

Na primeira etapa, onde é preciso fazer a síntese de áudio, foi utilizada a biblioteca libpd 1.4.6 onde esta extrai a funcionalidade do Pure Data e o torná disponível como um callback de processamento de áudio. A biblioteca libpd remove algumas funcionalidades como interface de usuário, tempo e capacidade de segmentação do Pure Data, tornando uma biblioteca mais flexível, sendo melhor utilizada nesta etapa para a geração de som e sintetização dos áudios recebidos e enviados.

A biblioteca libpd consegue a separação completa das preocupações entre desenvolvimento de áudio e de desenvolvimento de aplicativos, bem como um fluxo de trabalho suave. Com libpd, o protótipo é o código de produção (BRINKMANN, 2012), sendo mais eficiente para está aplicação, onde a partir da primeira etapa concluída pode se dar um foco maior para as demais.

Uma outra ferramenta analisada para a geração de som, foi utilizar classes nativas da linguagem Java, Sound Pool e Media Player são algumas dessas classes, Sound Pool é mais eficaz em relação ao Media Player, na geração de curtos sons e o Media Player e mais eficaz para grandes arquivos sonoros, mas esta aplicação necessita principalmente de sintetização de áudios, onde o libpd acaba sendo mais eficaz.

3.2 Segunda Etapa

Na segunda etapa, onde é preciso fazer a comunicação em rede da Orchidea envolveu duas questões principais: o método de endereçamento de rede a ser utilizado e a

estrutura de dados para o empacotamento das mensagens de rede.

Existem alguns métodos de endereçamento para transmissão de mensagens em rede como unicast(AALTONEN; KARVO; AALTO, 2002), broadcast(KAASHOEK et al., 1989) e multicast(COSTA,). Mensagens unicast são trocadas entre hosts específicos, como um computador pessoal e um servidor Web, mensagens broadcast são enviadas a todos os usuários de uma rede e mensagens multicast são enviadas a apenas uma parte específica de usuários em uma rede (CAMPOS,). Por esta razão, novas aplicações emergentes como vídeo conferência, computação distribuída, seminários e alguns aplicativos Android que necessitam do envio de dados para destinatários específicos em uma rede local são tipicamente processos que necessitam de comunicação Multicast.

Multicasting é um método de endereçamento de mensagens de rede para múltiplos destinos ao mesmo tempo onde, durante uma transmissão, o transmissor envia os pacotes de dados somente uma vez. Esta técnica diminui consideravelmente o tráfego em situações onde o mesmo pacote deve ser enviado a múltiplos destinatários.

Para o contexto deste trabalho, ao se analisar a aplicabilidade dos mecanismos de transmissão de dados em uma rede, o método de endereçamento multicast mostrou-se o mais adequado.

Quanto a estrutura de dados para o empacotamento das mensagens de rede, alguns formatos poderiam ser utilizados no contexto desta aplicação como XML, JSON, OSC e texto plano.

Apesar da popularidade dos formatos XML e JSON para aplicações web, o OSC (Open Sound Control) é um formato de mensagens focado no controle de aplicações músicais de tempo real(SCHMEDER; FREED; WESSEL, 2010). Este foco em comunicação tempo real evita a conversão de valores inteiros e ponto flutuantes em textos, como nas outras alternativas citadas, diminuindo com isto a largura de banda necessária para o envio de uma mensagem e aumentando com isto a velocidade de entrega de um pacote de rede.

3.3 Terceira Etapa

Na terceira etapa, onde é preciso desenvolver a interface de usuário, foi visto que a cada instrumento e tela desenvolvida no aplicativo, necessitará de uma activity diferente para cada uma desta, pois mesmo que as telas não tenham muita diferença gráfica entre uma é outra, suas tarefas serão diferentes, principalmente em cada instrumento, onde cada instrumento necessita de um patch desenvolvido em libpd 1.4.6 para a geração sonora, em relação as activities, estas foram implementadas utilizando XML, linguagem geralmente utilizada para a criação da interface de usuário.

4 Implementação

Para melhor visualização do que foi desenvolvido no aplicativo, foi elaborado um diagrama de classe que pode ser visto na figura 9, este foi escolhido por fornecer uma perspectiva do sistema de um ponto de vista externo. Colaborando com a implementação, pois como o desenvolvedor já sabe que classes criar e qual se relacionam com qual, tem-se uma maior facilidade de desenvolvimento.

Baseando-se na arquitetura proposta 2, foram criados objetos representando os componentes descritos na arquitetura, mas nesses objetos não há apenas a descrição dos componentes, mas também passa uma noção de como funcionara a relação entre estes no desenvolvimento do trabalho em um todo.

A implementação pode ser vista no diagrama de classes apresentado na Figura 9 que traz as classes que implementam os componentes descritos na arquitetura e a relação entre estas no desenvolvimento do trabalho.

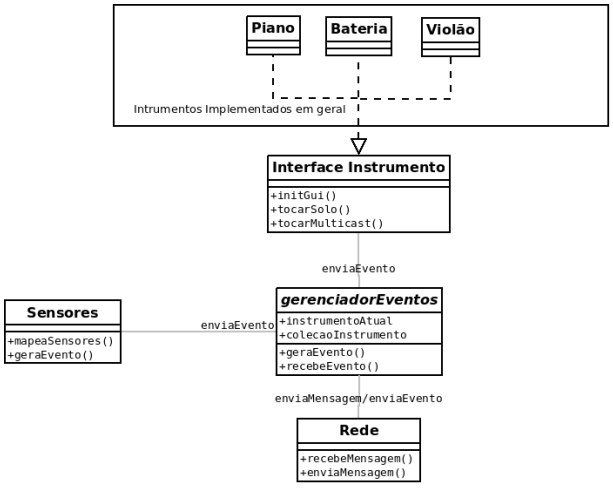

Figura 9: Diagrama de classe da Aplicação

Cada camada representada na arquitetura proposta é representada por um ou mais objetos. A camada de Interface de Usuário/IO está representada pelo objeto Sensores e Instrumento, a camada de Aplicação está representada pelo objeto gerenciadorEventos, e a camada de Comunicação é representada pelo objeto Rede.

A implementação foi feita para duas funcionalidades diferentes o usuário poder tocar individualmente apenas com o seu próprio “smarthphone“ e a implementação onde o usuário pode tocar com outros usuários em uma rede local, estas denominadas como Solo e Multicast da tela inicial do projeto.

4.1 Instrumento

Está classe Instrumento foi definida como sendo uma classe abstrata interface para diminuir o acoplamento e aumentar a coesão do projeto, classe está que ficou responsável por fazer a abstração do que significa um instrumento para o ambiente, instrumentos estes que foram desenvolvidos implementando os métodos initGui, tocarSolo e tocar.

O método initGui é o responsável por gerar a interface interface gráfica, uma destas interfaces podem ser vistas na figura 10 e fazer o mapeamento dos sensores para os botões desta interface.

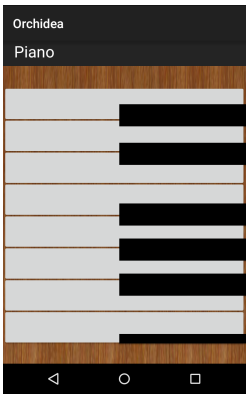

Figura 10: Interface do Piano

O método tocarSolo é o responsável por uma das funcionalidades do aplicativo que é fazer com que o usuário toque individualmente, esta funcionalidade foi implementada utilizando o libpd como sendo o responsável pela geração sonora, onde apenas foi necessário fazer uma chamada desta biblioteca, dado que o mapeamento da tecla já estava feito, sendo feita a chamada de tal no patch do pd utilizado no projeto.

O método tocar é o responsável pela outra funcionalidade do aplicativo que é fazer com que diversos usuários toquem utilizando uma rede local, simulando uma orquestra que é o objetivo do aplicativo, este método utilizada a classe Rede4.3, descrita mais a frente.

4.2 Aplicação

A Classe gerenciadorEventos pode ser vista como a classe principal da aplicação pois todas as ações que acontecem são supervisionadas ou impulsionadas por esta. Sua funcionalidade principal é supervisionar os eventos que são gerados pelos sensores ou recebidos pela rede, encaminhando estes eventos para os componentes responsáveis por refletir esta ação na aplicação. Esta classe está implementada de forma implícita pois o gerenciamento dos eventos ocorre em cada Instrumento da Orquestra.

4.3 Rede

A classe Rede é responsável por fazer o envio das mensagens de sons geradas para os demais dispositivos conectados a rede e também é responsável por receber as mensagens enviados por outros dispositivos conectados no ambiente, sendo que todos os instrumentos conectados tocam eles mesmo e os demais conectados. Para esta classe foram implementados os métodos sendMessage e receiveMessage.

O método sendMessage é o responsável por enviar a mensagem para a rede local, mensagem está enviada para um IP padrão “228.5.6.7“ e para a porta padrão “6789“, mensagens estas enviadas utilizando o formato OSC 1.4.9 da seguinte maneira: “/orch“ como sendo o namespace e dois argumentos “instrumento“ e “nota“, onde o instrumento é uma String contendo qual instrumento da orquestra irá tocar determinada nota e nota sendo um Float contendo o valor para determinada nota tocada, para uma melhor execução de determinada tarefa, foi utilizado o sistema de thread para o envio de tais mensagens, para um melhor desempenho do envio, para tratar um pouco do “delay“ da rede todos as mensagens são enviadas para a rede, inclusive as que o usuário atual envia, fazendo com que até as mensagens do usuário atual tenham o “delay“ da rede.

O método receiveMessage é o responsável por receber as mensagens da rede local, interpreta-las e designa-las a orquestra onde será tocada, para esta execução foi utilizada uma ferramenta proveniente no android, “Asynctask“ para melhorar o desempenho do recebimento, pois está ferramenta segue o mesmo conceito de uma thread.

5 Resultados

A proposta deste projeto traz uma visão de requisitos funcionais que a ferramenta proposta deveria ter, onde apenas expressava como ele vai funcionar, sem mais detalhes.

Na arquitetura foi feito uma análise desta proposta e apresentado um diagrama de camadas. A partir desta proposta arquitetural pode-se definir como seria dividido os problemas em um conjunto de componentes. Os componentes propostos foram divididos em classes e métodos no diagrama de classe da implementação, componentes estes que no primeiro momento foram vistos como classes, mas que foram transformados em algumas classes ou métodos de uma classe em um refinamento da proposta.

A decisão de projeto buscou recuperar a proposta inicial da ferramenta e trazer soluções de implementação para a Orchidea. Nesta etapa foram escolhidas as seguinte ferramentas, a) Android como sendo o sistema operacional a qual sera desenvolvida a aplicação, foi escolhido por ser uma plataforma aberta e bem documentada para os desenvolvedores; b) libpd como uma biblioteca para geração sonora e síntese, escolhida por desacoplar a síntese, geração de sons e mixagem de todos os instrumentos; c) Multicast para a transmissão de mensagens, por diminuir o tráfego na rede; d) OSC para o empacotamento de mensagens, pois é um formato de mensagens focado no controle de aplicações músicais de tempo real compatível com diversas ferramentas existentes para práticas músicais.

Após conceituar e implementar as ferramentas que foram utilizadas no projeto, pode-se notar que a biblioteca libpd 1.4.6 responsável pela síntese sonora, simplifica a alteração dos instrumentos e a inclusão de novos instrumentos e permite a colaboração com músicos e artistas que utilizam esta plataforma para criação de instrumentos e síntese.

Ao utilizar o OSC 1.4.9 e Multicast 1.4.5 permitiu a aplicação integrar a orquestra outros dispositivos que utilizam este tipo de empacotamento e endereçamento, como por exemplo utilizar um computador conectado a mesma rede para enviar as mensagens para os dispositivos móveis, exemplo este citado e analisado mais adiante.

Com isso temos que ao iniciar a aplicação o usuário se depara com as seguintes funcionalidades, “Solo“ e “Multicast“, que pode ser vista na imagem 11.

O usuário escolhendo qualquer uma das duas funcionalidades ele será redirecionado para uma tela contendo uma lista de instrumentos, que pode ser vista na imagem 12, ao escolher o instrumento que deseja tocar, este será redirecionado para a tela com a interface e funcionalidade do instrumento escolhido, um destes instrumentos pode ser visto na figura 10.

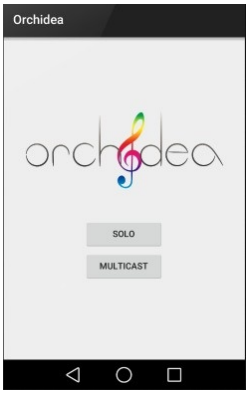

Figura 11: Tela Inicial

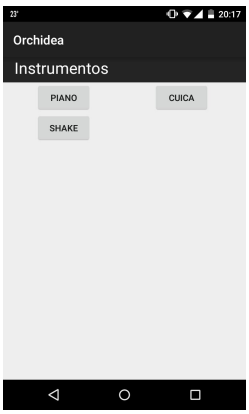

Figura 12: Escolha dos Instrumentos

Para analisar os resultados dessa aplicação utilizando a funcionalidade “Solo“, esta foi instalada em um aparelho contendo o sistema operacional Android, Nexus 4 contendo a seguinte especificação:

- Quad-Core 1.5 GHZ
- 2GB RAM
- Android 5.1.1

Ao utilizar a aplicação utilizando a funcionalidade “Solo“, o usuário não interage com nenhum outro dispositivo que é o objetivo desta aplicação, onde pode-se notar que som é apenas emitido pelo seu próprio dispositivo móvel, mas este acaba sendo mais eficiente em relação a outra funcionalidades, pois esta não há a necessidade de se conectar em uma rede local.

Para analisar os resultados dessa aplicação utilizando a funcionalidade “Multicast“, foi necessário instalar em um outro aparelho para fazer a interação entre os dois dispositivos, o Moto G 2ª geração contendo a seguinte especificação:

- Quad-Core 1.2 GHZ

- 1GB RAM
- Android 4.4.4

Estes conectados em um roteador TP-LINK, uma rede local doméstica, que está conectado a um modem ASUS.

Pode-se notar que utilizando a aplicação, o áudio emitido sai nos dois dispositivos simultaneamente de forma clara e limpa, cumprindo com o objetivo principal do projeto, fazendo com que os usuários possam interagir utilizando uma aplicação músical da maneira que lhes convir.

A aplicação utilizada nesta rede se mostrou eficiente, dado que aos usuários utilizarem os instrumentos implementados, o tempo de resposta dos dois aparelhos foi quase que imediato, sendo que aos ouvidos dos usuários foi imperceptível o “delay“ da rede. Mostrou se também versátil, pois cada usuário utilizou um instrumento para fazer os testes e o som emitido foi o mesmo nos dois dispositivos conectados.

Outra forma de utilizar esta aplicação é conectando um computador a mesma rede local e que possa fazer envio de mensagens utilizando o protocolo OSC 1.4.9 para os dispositivos móveis conectados.

Para esta analise foi utilizado um computador com as seguintes especificações:

- Intel Core i3 (3M Cache 1.80 GHz)
- 3GB RAM
- Ubuntu 12.04

As mensagens foram enviadas utilizando o terminal do sistema operacional Ubuntu, onde a mensagem foi enviada seguindo este padrão: oscsend osc.udp://228.5.6.7:6789 /orch sf “Instrumento“ “Nota“ esta mensagem é padrão e os únicos campos que podem ser alterados são “Instrumentos“ e “Nota“ que um referencia qual instrumento da orquestra irá tocar e o outro referencia qual nota irá tocar.

Pode-se notar que o áudio emitido sai nos dois dispositivos simultaneamente de forma clara e limpa, não apenas cumprindo com o objetivo principal do projeto mas também fazendo com que os usuários possam interagir utilizando não apenas dispositivos móveis mas também computadores portáteis e desktops.

Um exemplo do funcionando da aplicação Orquidea pode ser visto na Figura 13. Em que pode-se notar o funcionando desta para três dispositivos móveis e mais um computador conectados a uma mesma rede, sendo a dada no exemplo uma rede local.

Com isto, o objetivo do projeto foi alcançado, onde temos os conceitos de algumas ferramentas disponíveis sobre computação músical, a analise de qual ferramenta melhor

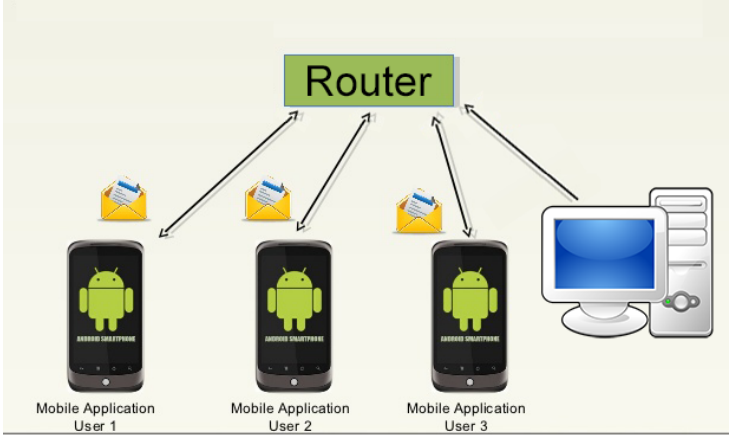

Figura 13: Exemplo funcionalidade Multicast

se encaixa no desenvolvimento do projeto e a implementação destas em um dispositivo móvel.

6 Considerações Finais

A possibilidade de integrar pessoas por meio de aplicações músicais para dispositivos móveis remete ao objetivo inicial destes aparelhos: a comunicação, mas amplia os limites desta comunicação permitindo e possibilitando a comunicação músical. Este documentou apresentou o projeto Orchidea, uma aplicação para produção músical colaborativa / cooperativa por meio de uma orquestra de dispositivos Android.

Foi apresentado a proposta inicial da ferramenta, o modelo arquitetural proposto, a implementação desta arquitetura e as decisões de projeto tomadas para a criação e implementação deste aplicativo. Entre as decisões de projeto tomadas estão a utilização de XML para a implementação da GUI, libpd como a ferramenta para a síntese e geração sonora, OSC como a ferramenta para o empacotamento das mensagens e o protocolo Multicast para o envio destas mensagens. Acreditamos que a ferramenta desenvolvida atende aos requisitos propostos e permite a criação músical colaborativa cumprindo com o objetivo principal desta aplicação.

Além disto, o desenvolvimento desta aplicação permitirá uma pesquisa interdisciplinar envolvendo musicista e leigos na criação de uma orquestra de celulares na Universidade Federal de São João Del Rei como um projeto de pesquisa de parceria entre o Departamento de Ciência da Computação e o Departamento de Música.

6.1 Trabalhos Futuros

Contudo, esta aplicação pode ter algumas melhorias, sendo uma destas a utilização de outros sensores disponíveis nos “smartphones“, como o acelerômetro, giroscópio, não apenas os sensores, pode-se utilizar também as câmeras dos dispositivos móveis, como foi citado no trabalho relacionado 1.3.

Outra melhoria disponível é fazer com que a aplicação seja mais abrangente, ou seja deixar de funcionar apenas para rede local e ter uma funcionalidade onde esta funciona para toda a rede, contendo um servidor centralizada para receber e redirecionar as mensagens enviadas pelos usuários.

Referências

AALTONEN, J.; KARVO, J.; AALTO, S. Multicasting vs. unicasting in mobile communication systems. In: *Proceedings of the 5th ACM International Workshop on Wireless Mobile Multimedia*. New York, NY, USA: ACM, 2002. (WOWMOM '02), p. 104–108. ISBN 1-58113-474-6. Disponível em: <http://doi.acm.org/10.1145/570790.570808>. Citado 2 vezes nas páginas 16 e 26.

AMALFITANO, D.; FASOLINO, A. R.; TRAMONTANA, P. A gui crawling-based technique for android mobile application testing. In: *ICST Workshops*. IEEE Computer Society, 2011. p. 252–261. Disponível em: <http://dblp.uni-trier.de/db/conf/icst-/icstw2011.htmlAmalfitanoFT11>. Citado na página 16.

BRINKMANN, P. *Making Musical Apps*. O'Reilly, 2012. (Oreilly and Associate Series). ISBN 9781449314903. Disponível em: <https://books.google.com.br-/books?id=6uUNZ9yH7wsC>. Citado na página 25.

BRINKMANN PETER KIRN, R. L. C. M. M. R. H.-C. S. P. Embedding pure data with libpd. Citado na página 17.

BUKVIC, I. et al. Introducing l2ork : Linux laptop orchestra. In: BEILHARZ, K. et al. (Ed.). *Proceedings of the International Conference on New Interfaces for Musical Expression*. Sydney, Australia: [s.n.], 2010. p. 170–173. Disponível em: <http://www.nime.org/proceedings/2010/nime2010_170.pdf>. Citado na página 14.

BURNETTE, E. *Hello, Android: Introducing Google's Mobile Development Platform*. Pragmatic Bookshelf, 2010. (Java Programming / Telephony). ISBN 9781934356562. Disponível em: <https://books.google.com.br/books?id=cPZbRQAACAAJ>. Citado na página 15.

CAMPOS, T. de L. Desenvolvimento de uma aplicaÇÃo de chat utilizando multicast e criptografia simÉtrica. Citado 2 vezes nas páginas 17 e 26.

COSTA, O. C. M. B. D. L. H. M. K. Roteamento multicast na internet. Citado 2 vezes nas páginas 17 e 26.

DANNENBERG, R.; SCAVACO; ANG I. AVRAMOVIC, B. A. J. B. E. B. D. D. J. G. R. H. C. J. U. K. D. M. T. M. M. R. D. T. A. Y. E. The carnegie mellon laptop orchestra. In: *Proceedings of the 2007 International Computer Music Conference, vol II*. [S.l.]: The International Computer Music Association, 2007. p. 340–343. URL=http://ctp.di.fct.unl.pt/ sc/publicacoes/icmc07laptop.pdf. Citado na página 14.

ESSL, G.; ROHS, M. Shamus – a sensor-based integrated mobile phone instrument. In: *IN PROCEEDINGS OF THE INTERNATIONAL COMPUTER MUSIC CONFERENCE (ICMC*. [S.l.: s.n.], 2007. p. 27–31. Citado na página 13.

KAASHOEK, M. F. et al. An efficient reliable broadcast protocol. *OPERATING SYSTEMS REVIEW*, v. 23, p. 5–19, 1989. Citado 2 vezes nas páginas 16 e 26.

LEE, S. W. et al. Collaborative musical improvisation in a laptop ensemble with lolc. In: GOEL, A. K. et al. (Ed.). *Creativity Cognition*. ACM, 2011. p. 361–362. ISBN 978-1-4503-0820-5. Disponível em: <http://dblp.uni-trier.de/db/conf/candc/candc2011-.htmlLeeFCYT11>. Citado na página 14.

OH, J. et al. Evolving the mobile phone orchestra. In: BEILHARZ, K. et al. (Ed.). *Proceedings of the International Conference on New Interfaces for Musical Expression*. Sydney, Australia: [s.n.], 2010. p. 82–87. Disponível em: <http://www.nime.org-/proceedings/2010/nime2010_082.pdf>. Citado na página 12.

PEREIRA, L.; SILVA, M. D. *Android para Desenvolvedores*. BRAS-PORT. ISBN 9788574524054. Disponível em: <https://books.google.com.br-/books?id=8u9wJowXfdUC>. Citado na página 15.

PUCKETTE, M. Pure data: another integrated computer music environment. In: . [S.l.: s.n.]. Citado na página 18.

SCHIEMER, G.; HAVRYLIV, M. Pocket gamelan: A pure data interface for mobile phones. In: *Proceedings of the 2005 Conference on New Interfaces for Musical Expression*. Singapore, Singapore: National University of Singapore, 2005. (NIME '05), p. 156–159. Disponível em: <http://dl.acm.org/citation.cfm?id=1085939.1085982>. Citado na página 12.

SCHMEDER, A.; FREED, A.; WESSEL, D. Best practices for open sound control. In: *Linux Audio Conference*. Utrecht, NL: [s.n.], 2010. Citado 2 vezes nas páginas 18 e 26.

TANAKA, A. Mobile music making. In: NAGASHIMA, Y.; ITO, Y.; FURUTA, Y. (Ed.). *Proceedings of the International Conference on New Interfaces for Musical Expression*. Hamamatsu, Japan: [s.n.], 2004. p. 154–156. Disponível em: <http://www.nime.org/proceedings/2004/nime2004_154.pdf>. Citado na página 13.

TRUEMAN, D. Why a laptop orchestra? *Organised Sound*, v. 12, p. 171–179, 8 2007. ISSN 1469-8153. Disponível em: <http://journals.cambridge.org/article_S135577180700180X>. Citado na página 14.

WANG, G. et al. Stanford laptop orchestra (slork). In: *International Computer Music Conference*. Montreal: [s.n.], 2009. Disponível em: <http://ccrma.stanford.edu/~ge-/publish/slork-icmc2009.pdf>. Citado na página 14.

WANG, G.; ESSL, G.; PENTTINEN, H. Do mobile phones dream of electric orchestras? In: . [S.l.: s.n.], 2008. Citado na página 13.

XU, Z.; BAI, K.; ZHU, S. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In: *Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks*. New York, NY, USA: ACM, 2012. (WISEC '12), p. 113–124. ISBN 978-1-4503-1265-3. Disponível em: <http://doi.acm.org/10.1145/2185448.2185465>. Citado na página 19.